The I/O benchmarks shall be run or projected for all offered subsystems including upgrades. An updated dudley.csh has been supplied within a GNU-zipped tar file (iobenchv2.tgz) on the rdhpcs website. The new I/O benchmark suite (iobenchv2.tgz) is completely self-contained and replaces the contents of the CD containing the I/O benchmark previously distributed. It should be noted that the Linux kernel build has been removed from the I/O benchmark and has been replaced by a build of Python. Some of the scripts use the command “date +%s” to get time in seconds; the Offeror is free to substitute this command with lequivalent functionality.

Timing results shall be presented for all I/O benchmarks running concurrently. Do to start up skew, it may be necessary to drop the first sample and one or more of the last samples from total samples. The Offeror shall specify the sampling methodology. Within the precision of clock synchronization, the Offeror shall use the date command (or equivalent functionality) to ensure that the entire benchmark sample runs while all other instances are running. The Offeror shall average within an instance and then across instances to calculate an overall average.

The first system test is designed to determine the overall I/O bandwidth to the offered file systems (see I/O spreadsheet). The dudley.csh script is located in the ddbench subdirectory in the tar file supplied on the rdhpcs web site. Note that the rdhpcs web site version has changed from the original dudley.csh script supplied on the CD. The script takes two arguments: a path where the scratch files are written and read and the number of times the lmdd test is to repeat (10 is the default).

The dudley.csh script uses the lmdd application whose source is available in the lmdd subdirectory. (The lmdd application is built by simply compiling the two C source files in that directory with a resulting executable named lmdd.) If any controllers, communications connections, file servers, metadata servers or disk subsystems are shared, the Offeror shall describe the performance impacts of the sharing. Read and write performance shall be presented separately in megabytes per second as reported by lmdd (e.g. 127.2/131.4 means read bandwidth is 127.2 MB/s and write bandwidth is 131.4 MB/s).

The second file system test is designed to test metadata server performance for the offered file systems. The Python build test is contained in the pybuild directory in the tar file supplied on the rdhpcs web site. The pybuild directory contains a script named “build.csh”. This script takes two arguments: a path where the python build will take place and the number of times the python build is to repeat (10 is the default). Each instance of build.csh must have a unique path in which to build. The environment variable “truesize” contains the expected size of the Python binary. The build.csh displays this size if it does not match the expected size. There are many reasons why executables may differ in size and

thus in general, the messages generated may be ignored. The build.csh script should be run on all HPCS cores proposed that allow compilation.

Results for bandwidths should be presented as averages rather than sums when multiple simultaneous instances are required in the worksheet. Averages for elapsed time for the Python build should also be presented (“Total time” output from build.scr)

Additionally, the iobenchv2.tgz file obtained from the rdphcs web site also contains three codes that may be run at the Government's discretion during acceptance testing. This is in accordance with the first paragraph of E.2.2 of the RFP.

The parwrite and doublewrite codes are tests that have exposed problems in our existing system(s). Parwrite simulates a code that writes to disk simultaneously from all MPI processes. This has been known to cause large loads on a single I/O server depending on the I/O architecture. Doublewrite exposed an error in the Linux implementation of the XFS file system. This was a race condition that has subsequently been patched in the kernel. This will be run on shared file systems to ascertain whether a similar error exists in the supplied file systems.

Sanity is a diagnostic code which ensures that all processes can communicate with each other. It will be used as a simple application in testing the batch system and/or job launch mechanism.